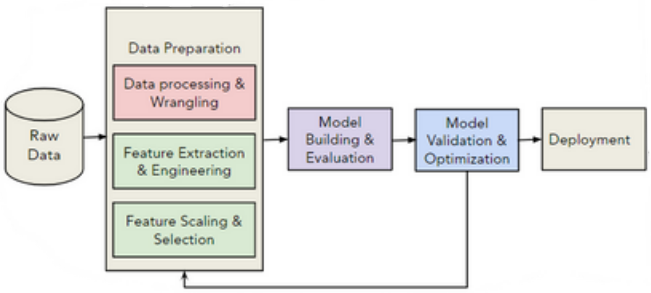
SLR cheat sheet

Steps in ML Model deployment



Simple Linear Regression formula

y = β0 + β1x + ε

Where:

- β0 = y intercept
- β1 = slope
- x = set of values taken by independent variable X
- y = target/dependent variable
- ε = random error term

Note: Error term also called residual, it represents the distance of the observed value from the value predicted by regression line.

ε = yactual - ypredicted

Multiple linear regression

y = β0 + β1x1 + β2x2 + β3x3 + ... +βnxn + ε

β0, β1, β2, β3, ..., βn are the parameters of the linear regression model with n

independent variables

y = set of values taken by dependent variable Y

xi = set of values taken by independent variable Xi , i ∈ [1,n] β0 = y intercept

βi = beta coefficient for the ith independent variable Xi, i ∈ [1,n]

ε = random error component

Ordinary least Squares

$$min \sum_{i=1}^{n} (y_i - \beta_i x_i)^2$$

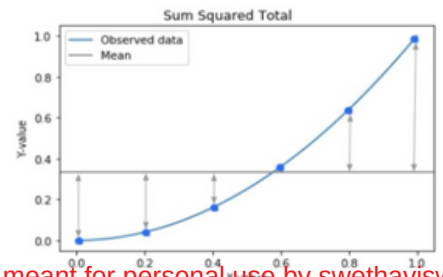
- The ordinary least square method is used to find the best fit line for given data
- This method aims at minimizing the sum of squares of the error terms, that is, it determines those values of **β0 and β1 at which the error terms are minimum**

Measures of variation

Sum of squares total (SST)

It can be seen as the total variation of the response variable about its mean value.

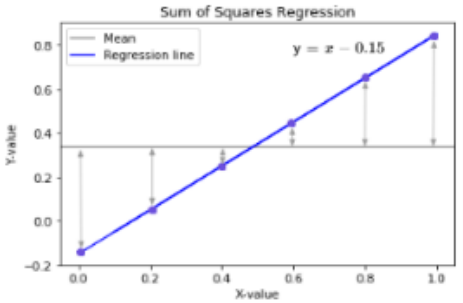
$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2$$



Sum of Squares Regression (SSR)

SSR is the measure of variability in the response variable considering the effect of predictor variable . It is the explained variation

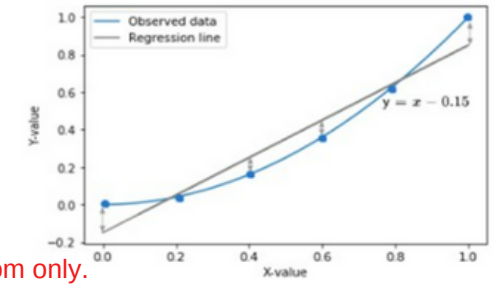
$$SSR = \sum_{i=1}^{n} (\hat{y} - \bar{y})^2$$



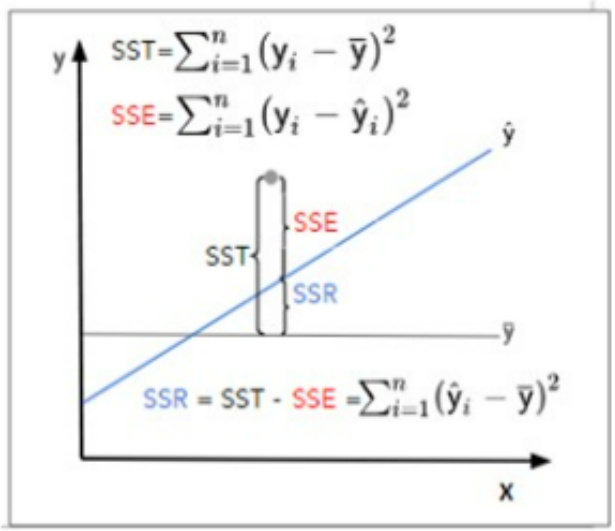
Sum of squares of error (SSE)

The sum of squares of error (SSE) is the sum of squared differences between observed response variable and its predicted value

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y})^2$$



This file is meant for personal use by swethaviswanathan.r@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.



Measure of unexplained variation

- Standard error of estimate is a measure of the unexplained variance
- Smaller value of standard error of estimate indicates a better model

$$min \sum_{i=1}^{n}(y_i - \beta_i x_i)^2$$

n = sample size
k = number of parameter estimates (β0,β1)

Measure of explained variation

R2 also called the coefficient of determination gives total percentage of variation in Y that is explained by predictor variable.

$$R^2 = \frac{\text{Explained variation}}{\text{Total variation}} = \frac{\text{SSR}}{\text{SST}}$$

$$0 \leq R^2 \leq 1$$

$$R^2 = 1 - \frac{SSE}{SST}$$

T-test for significance in SLR

H0: The parameter β is not significant
H1: The parameter β is significant

T-test for slope in SLR

H0: There is no relationship between variables X and Y
H1: There is relationship between variables X and Y

The t test for intercept

H0: The parameter β0 is not significant
H1: The parameter β0 is significant

ANOVA for regression

H0: The regression model is not significant
H1: The regression model is significant

R2 vs. Adjusted R2

- The value of R Squared never decreases. If we add new independent variables, then the value of R Squared increases. It cannot show the effect of adding a bad or insignificant variable
- As compared to the R-Squared value, Adj. R-Squared has an ability to decrease whenever a bad or an insignificant variables are added to the model. Thus we get an accurate evaluation.

Assumptions of Linear Regression

1. **The dependent variable must be numeric**
   - check the data type of target variable

2. Predictors must not show multicollinearity

- Multicollinearity arises when the **independent variables have high correlation** among each other.
- Detection of multicollinearity:
  - Determinant of correlation matrix
    - Let D be the determinant of correlation matrix.
      - 0 < D < 1

D=0	High multicollinearity
D=1	No multicollinearity

  - Condition Number (CN)
    - present in OLS summary

CN > 1000	Severe multicollinearity
100 < CN < 1000	Moderate multicollinearity
100 < CN	No multicollinearity

  - Correlation matrix
    - .corr()
  - Variance Inflation Number (VIF)

$$VIF = \frac{1}{1-R^2}$$

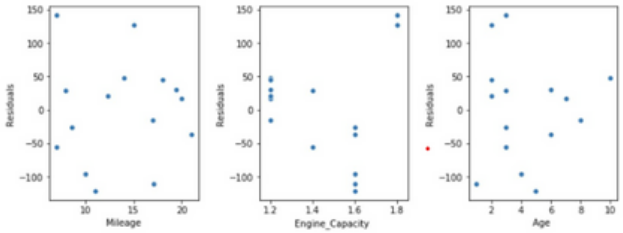
```
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Drop Dependent Variable a.k.a Target Variable
x = premium.drop("Premium", axis = 1)

# create an empty dataframe to store the VIF for each variable
vif = pd.DataFrame()


# calculate VIF using list comprehension
# use for loop to access each variable
# calculate VIF for each variable and create a column 'VIF_Factor' to store the values
vif["VIF_Factor"] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]

# create a column of variable names
vif["Features"] = x.columns

# sort the dataframe based on the values of VIF_Factor in descending order
# 'ascending = False' sorts the data in descending order
# 'reset_index' resets the index of the dataframe
# 'drop = True' drops the previous index
vif.sort_values('VIF_Factor', ascending = False).reset_index(drop = True)
```

3.Linear relationship between dependent and independent variables

- This can be checked by plotting a scatter plot of residuals vs predictors

This file is meant for personal use by swethaviswanathan.r@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

- A scatter plot depicting no pattern indicates that the variable has a linear relationship with the response variable



4.Independence of observations should exist(ie. Absence of Auto correlation

- Assumption of autocorrelation is violated **when residuals are correlated within themselves**, ie they are serially correlated
- **Durbin - Watson Test (present in OLS summary)**
  - H0: The error terms are not autocorrelated
  - H1: The error terms are autocorrelated

Value	Interpretation
0 < d <2	Positive autocorrelation
d = 2	No autocorrelation
2 < d < 4	Negative autocorrelation

5.The error terms should be homoscedastic

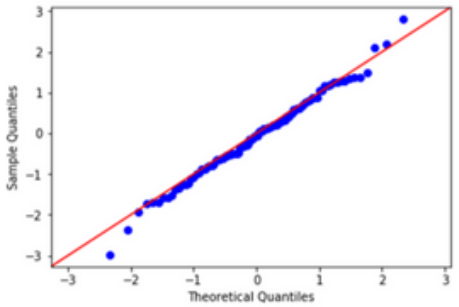
- Variance of the residual is assumed to be independent of the explanatory variables
- Heteroscedasticity: non-constant variance of residuals
- Statistical tests:
  - Goldfeld Quandt test
    - H0: The errors terms are homoskedastic
    - H1: The errors terms are heteroskedastic
  - Breusch Pagan test
    - H0: The errors terms are homoskedastic
    - H1: The errors terms are heteroskedastic

```
import statsmodels.stats.api as gq
gq.het_goldfeldquandt(model.resid,predictors)
```

6.The error terms should follow a normal distribution

Normality test:

- Quantile-Quantile Plot



- Jarque-Bera (JB) Test
  - H0: Skewness (S) = 0 and Kurtosis (K) = 0
  - H1: Skewness (S) ≠ 0 and Kurtosis (K) ≠ 0

```
scipy.stats.jarque_bera(model.resid)
```

- Shapiro-Wilk Test
  - H0: The data is normally distributed
  - H1: The data is not normally distributed

```
stats.shapiro(model.resid)
```

Feature Transformation

- Incase of skewed (predictor and/or dependent) variable, we transform it to reduce the skewness.
- If the assumptions of linear regression are not met, transformation of skewed target variable can be used for making the error terms more compatible to the assumptions
- Transformation methods
  - Log-Transform(postivie values only)
    - data_log = np.log(data)
  - Squareroot Transform(positive and zero values)
    - data_log = np.sqrt(data)
  - Yeo-Johnson(positive zero and negative values)

```
from scipy import stat
transformed_data = st
```

  - Reciprocal Transformation(positive and negative values)

```
transformed_data = np.reciprocal(data)
```

  - Exponential Transformation(reverse of log)

```
transformed_data = np.exp(data)
```

Feature Scaling

It is a technique used to transform the data into a common scale

Feature Scaling methods:

- Standardization
  - Standardization rescales the feature such that it has mean 0 and unit variance
  - from sklearn.preprocessing import StandardScaler
  - scaler = StandardScaler()
  - data_scaled=scaler.fit_transform(data)

$$x' = \frac{x - \bar{x}}{\sigma}$$

- Normalization
  - Normalization is the process of rescaling features in the range 0 to 1
  - from sklearn.preprocessing import MinMaxScaler
  - scaler = MinMaxScaler()
  - data_scaled=scaler.fit_transform(data)

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

This file is meant for personal use by swethaviswanathan.r@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Feature Selection

- Feature selection is the process of including the significant features in the model
- This can be achieved by:
  - Forward selection method
  - Backward elimination method
  - Stepwise method

- **Forward Selection**

```
from mlxtend.feature_selection import SequentialFeatureSelector as sfs

Ridgelinreg_forward = sfs(estimator = model, k_features, forward = True
```

- **Backward Elimination**

```
from mlxtend.feature_selection import SequentialFeatureSelector as sfs

linreg_backward = sfs(estimator = model, k_features, forward = False)
```

- **Recursive Feature Elimination**

```
from sklearn.feature_selection import RFE

rfe_model = RFE(estimator = model, n_features_to_select)
```

Bias and Variance

- Model which doesn't perform well on Train data itself will have High Bias
- Model which performs well on train data but doesnt perform well on test data will have High Variance

Bias

- If the **model is too simple** it will have a **high bias and low variance**
- Such a model will give not perfectly accurate predictions, but the predictions will be consistent
- The model will not be flexible enough to learn from majority of given data, this is termed as underfitting

Variance

- If the **model is too complex** it will have a **low bias and high variance**
- Such a model will give accurate predictions but inconsistently
- The high variance indicates it will have a much better fit on the train data compared to the test data, this is termed as overfitting

Linear Regression

#Using sklearn

```
from sklearn.linear_model import LinearRegression
model=LinearRegression().fit(Xtrain,ytrain)
```

Or

#Using statmodels

```
import statsmodels.api as sm
model=sm.OLS(ytrain,sm.add_constant(Xtrain))
model.summary() #to get ols summary
```

Model Prediction

```
ypred=model.predict(Xtest)
```

Model Evaluation

R-Squared

```
from sklearn.metrics import r2_score
r_sq = r2_score(y_test, y_pred)
```

Adjusted R-Squared

```
n = number of observations
k = number of columns (including intercept)
adj_r_squared = 1 - (((1 - r_sq) * (n - 1)) / (n - k - 1))
```

Mean Squared Error

```
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test, y_pred)
```

Root Mean Squared Error

```
RMSE = np.sqrt(MSE)
```

Mean Absolute Error

```
from sklearn.metrics import mean_absolute_error
MAE = mean_absolute_error(y_test, y_pred)
```

Mean Absolute Percentage Error

```
MAPE = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
```

This file is meant for personal use by swethaviswanathan.r@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Model Validation

The model validation methods use test data to validate the model built using train data

Cross_val_score

This method is known as two fold cross validation

Here, each observation is used exactly once for training and once for testing

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(estimator = LinearRegression(), X = X_train, y = y_train, cv, scoring)
```

K-Fold CV

Here, each observation is used exactly k times for training and exactly once for testing

```
from sklearn.model_selection import KFold
kf = KFold(n_splits)
```

Leave One Out CV

It is a special case of k - fold cross validation method. Instead of subsetting the data, at every run one observation is considered as the test set

```
from sklearn.model_selection import LeaveOneOut
loocv = LeaveOneOut()
```

Grid Search

The estimates of parameters are usually estimated from the data

```
from sklearn.model_selection import GridSearchCV
grid_model = GridSearchCV(estimator, param_grid , cv )
```

Cost Function

- A cost function tells how good the model performs at making predictions for a given set of parameters
- Cost function = Loss function = Error function

$$Error = \sum_{i=1}^{n} (y_{act} - y_{pred})^2$$

Note: where yact is the actual value and ypred is the predicted value.

Gradient Descent

- The gradient descent is an optimization technique which finds the parameters such that the error term is minimum.
- It is an iterative method which converges to the optimum solution.
- It takes large steps when it is away from the solution and takes smaller steps closer to the optimal solution.
- The estimates of the parameter are updated at every iteration.

```
from sklearn.linear_model import SGDRegressor
sgd = SGDRegressor()
```

Regularization

- Regularization refers to the modifications we make to a learning algorithm, that help in reducing its generalization error but not its training error
- Regularization adds a cost function such tha higher variance receives a larger penalty
- It chooses a model with smaller parameter values (i.e. shrunk coefficients) that has less error

$$\text{Loss function}_{\text{regularization}} = \text{Loss function}_{\text{ols}} + \text{Penalty term}$$

Ridge Regression

- Ridge regression uses squared L-2 norm regularization i.e it adds a squared L-2 penalty
- It diminishes the insignificant predictors but does not completely eliminate them

```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha)
```

Lasso Regression

- Lasso regression uses L-1 norm regularization i.e it adds a L-1 penalty
- It extinguishes the insignificant predictors

```
from sklearn.linear_model import Lasso lasso = Lasso(alpha)
```

Elastic-net regression

- Elastic-net regression uses both L-1 and L-2 norm regularization
- Elastic-net regression is the combination of lasso and ridge regression

```
from sklearn.linear_model import ElasticNet
enet = ElasticNet(alpha, l1_ratio)
```

This file is meant for personal use by swethaviswanathan.r@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.